시간 적응 기법 적용을 위한 시스템 미분 방정식의 분석

넥스트폼 기술 연구소
경태윤

2020.10.20.

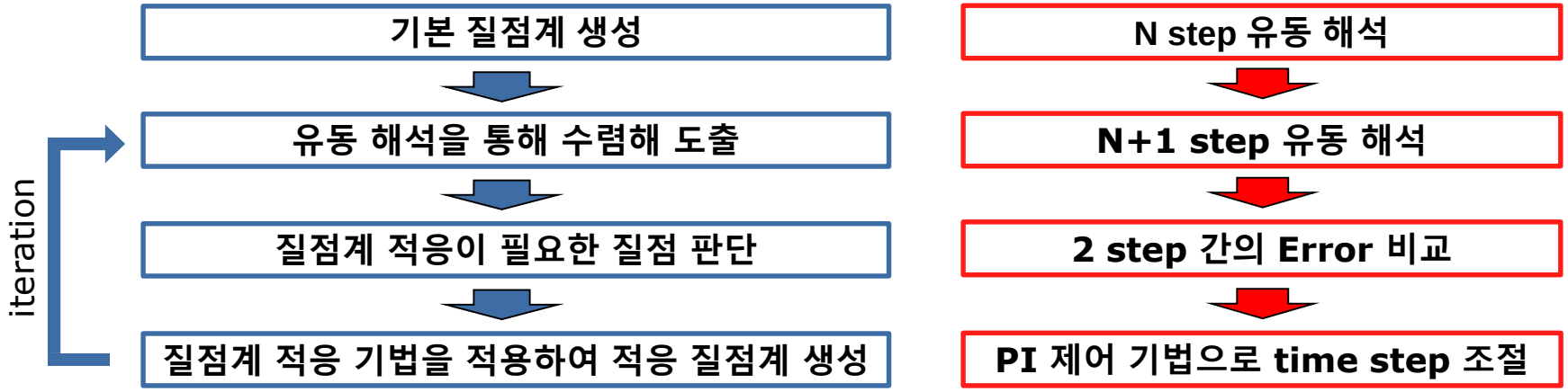
목 차

목 차

Adaptation 기법

Adaptation 개요

- 비정상 유동 해석의 정확도와 효율성을 동시에 높이기 위해 질점계 적응 기법(Spatial Adaptation)과 Timestep 조절(Time Adaptation)
  - Spatial Adaptation
    - 유동 해석 수렴해를 기반으로 질점계 적응이 필요한 공간을 파악하여 수행
    - Octree 기반의 질점계 – 규칙성을 이용하여 질점 추가/제거 용이
  - Time Adaptation
    - 비정상 유동 해석은 적절한 Time step 선정이 매우 중요
    - Physical Error의 경향을 분석하여 Time step을 자동적으로 조절



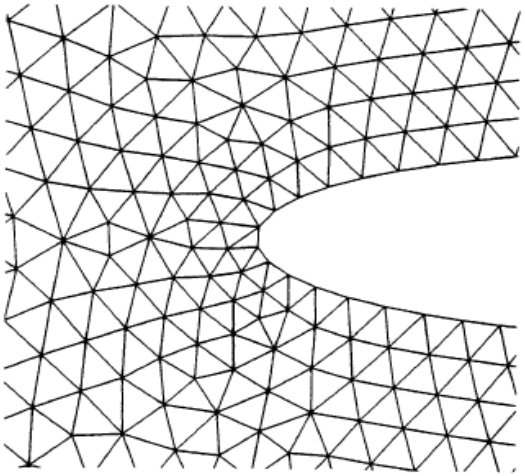
무격자 기법

◆ 무격자 해석 기술 소개

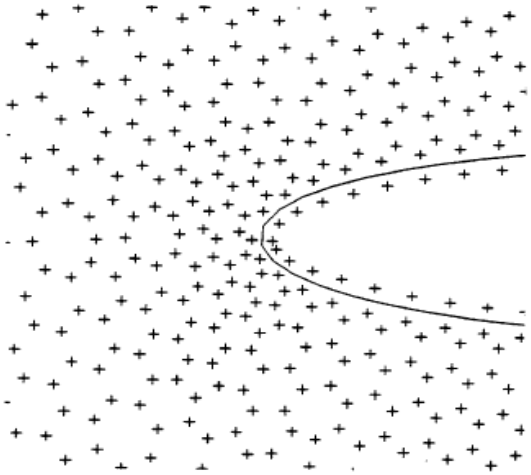
- 무격자 해석 기술
  - 격자의 개념을 사용하지 않고, 질점만을 이용하여 수치 해석을 하는 방법
  - 볼륨 격자의 개념이 없기 때문에 유동 계산 영역 생성이 유연 → 자동화 가능
  - 따라서 기존 격자 기반의 해석 기법에서 발생하는 과도한 전처리 작업시간 단축 가능

복잡한 형상이나 다물체 간 상대 운동이 존재하는 문제

→ 질점을 자동으로 생성하여 해석



Unstructured mesh



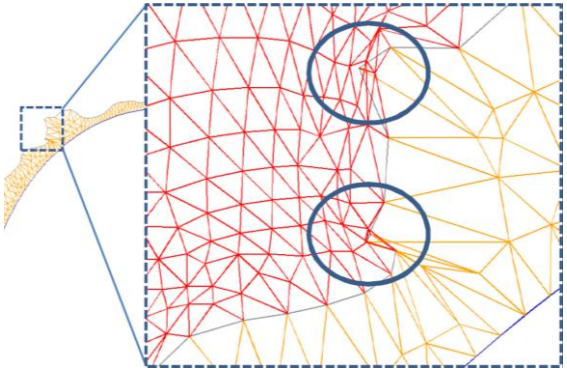
Meshless point system

무격자 기법

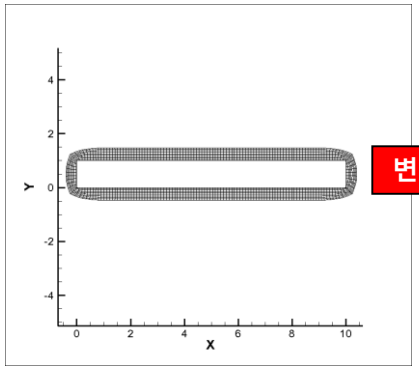
◆ 무격자 해석 기술 소개

- 격자계와 무격자계 비교
  - 격자계는 물체가 과도하게 이동하거나 변형되는 경우, 격자 꼬임 현상이 발생 가능
    - 격자의 재생성 과정에서 많은 시간이 소모
  - 이에 반해 무격자계는 격자 개념이 없이 점의 연결 정보만 이용하기 때문에 유동 계산 영역을 재생성하지 않고 해석 가능
  - 따라서 무격자 해석 기법은 에어백 전개와 같은 물체가 붙어 있는 상태에서 대변형이 존재하여 유동 계산 영역을 해석 중간에 생성해야하는 격자 기반의 해석 기법보다 자동화에 유리

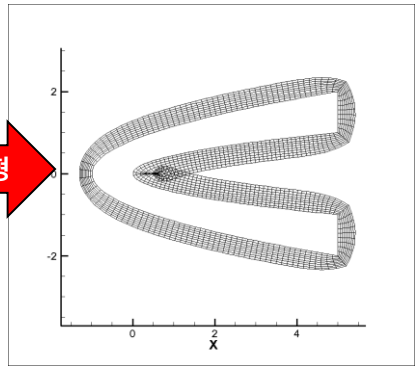
    → 수월하게 해석 가능



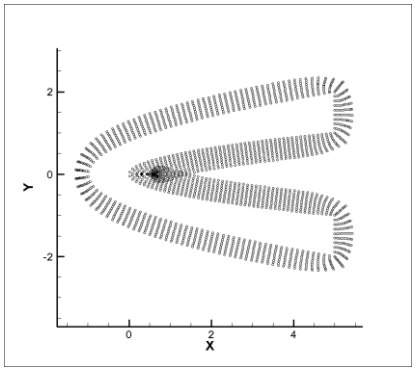
격자 꼬임 현상 예시



변형 전



변형 후
(격자 재생성 필요)

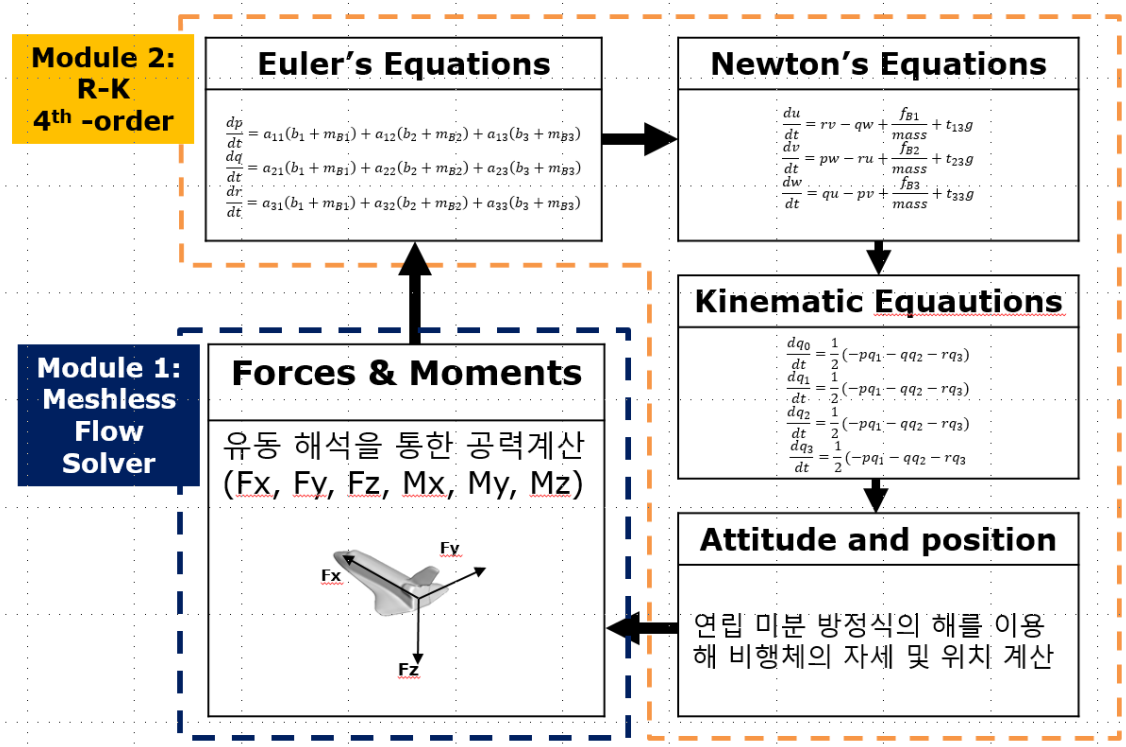


변형 후
(해석 가능)

이동 물체 해석 기법

◆ 무격자 질점 이동 기법

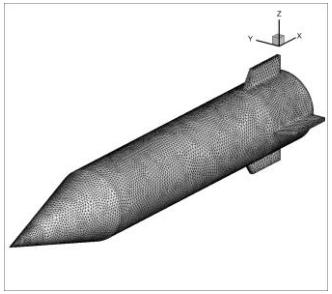
- 6 자유도의 움직임을 고려한 비정상 유동 해석
  - 실제 사용되는 비행체들의 움직임을 모사하기 위한 6자유도를 고려한 궤도 계산 기법을 무격자 코드에 적용
  - Translational, Attitude motion 모사를 위한 Newton's law 와 Euler's Law를 이용하여 궤도 해석



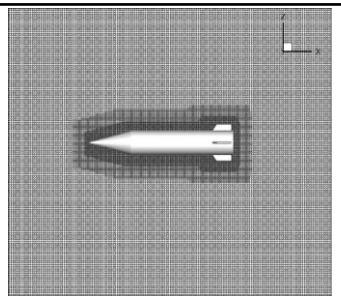
프로그램 구조

무격자 해석 프로그램

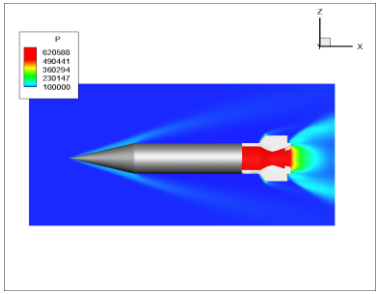
CAD 파일로 부터 표면 격자 자동 생성



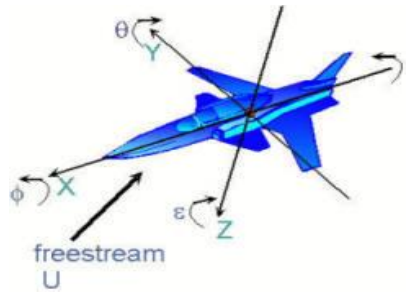
공간 질점 생성 및 Connectivity 구성



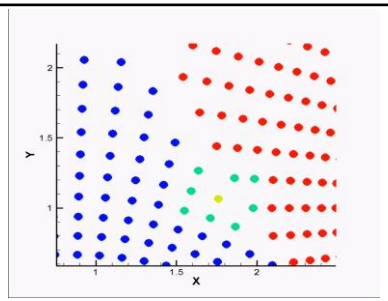
유동 해석 수행



6자유도 Dynamic Eq.으로 움직임 구현



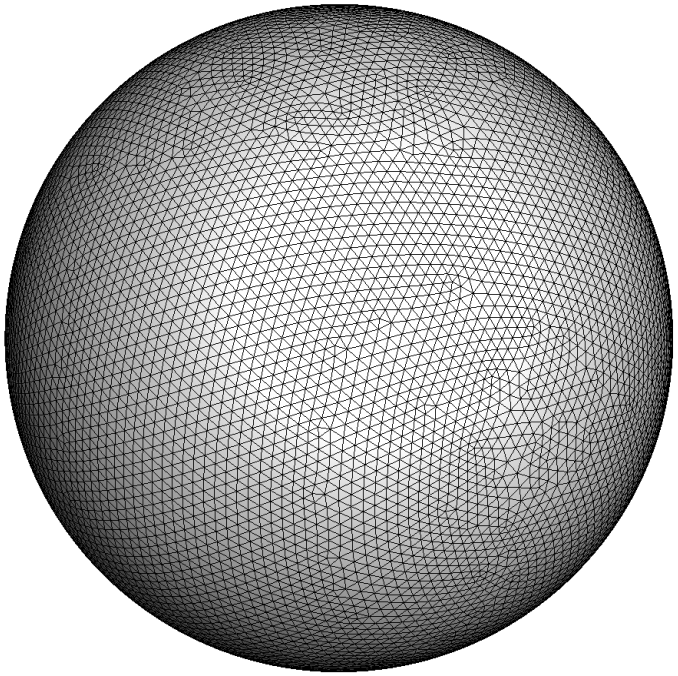
질점 이동 및 Connectivity 재구성



목 차

공간 질점계 적응 기법

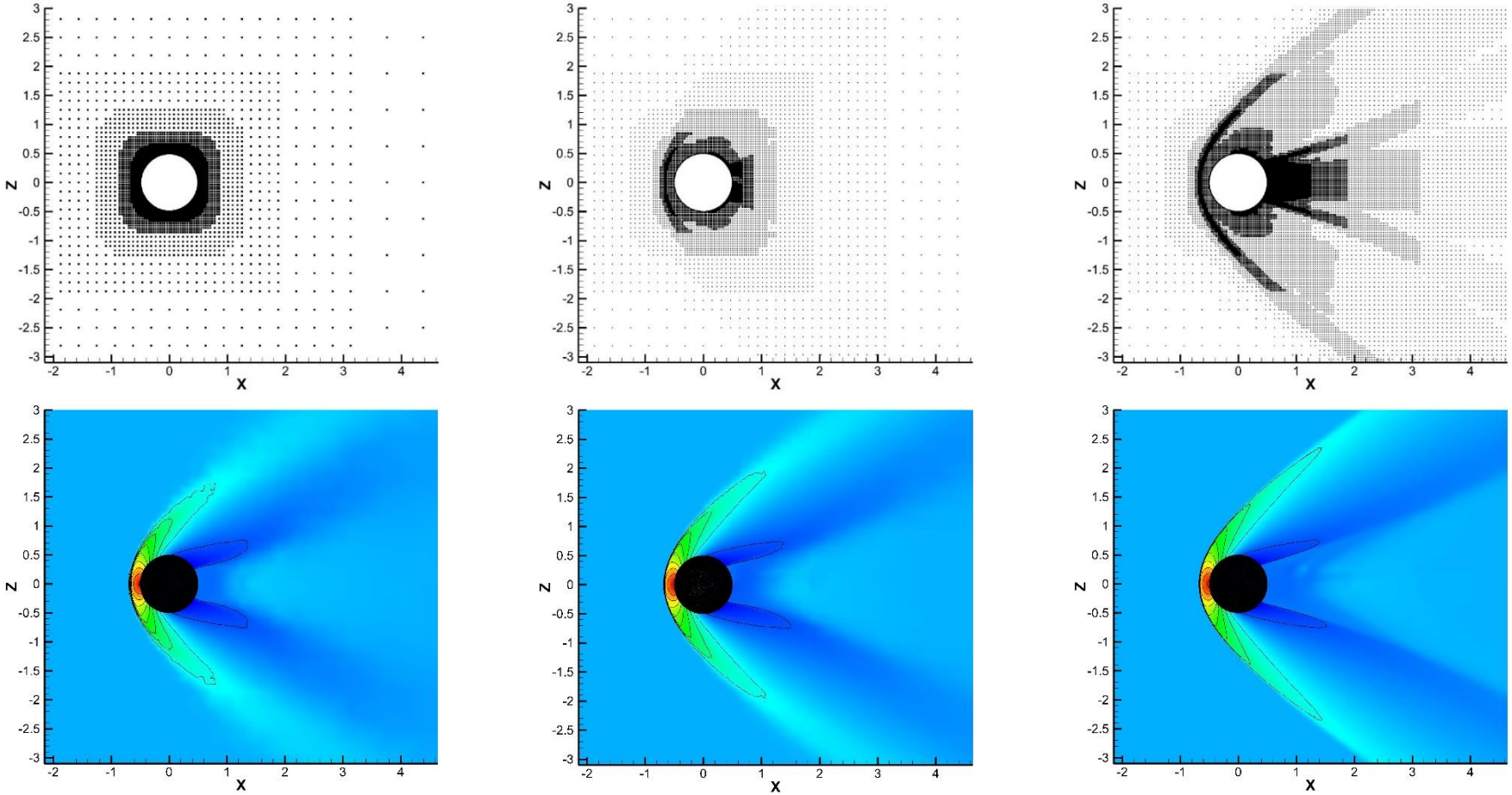
- Sphere 해석에서의 질점계 적응 기법
  - 해석 방정식 : Euler Equation
  - 공간 차분 : M-AUSMPW+
  - 공간 내삽 : 3rd MLP
  - 시간 적분 : LU-SGS & Dual Time Stepping



Operation condition	AOA (Deg)	자유류 속도 (Ma.)	온도(K)
	0.0	2.0	300.0

공간 질점계 적응 기법

◆ Sphere 해석에서의 질점계 적응 기법



<0 level adaptation>

313,532 개

<1 level adaptation>

783,495 개

<3 level adaptation>

3,629,915 개

목 차

시간 적응 기법

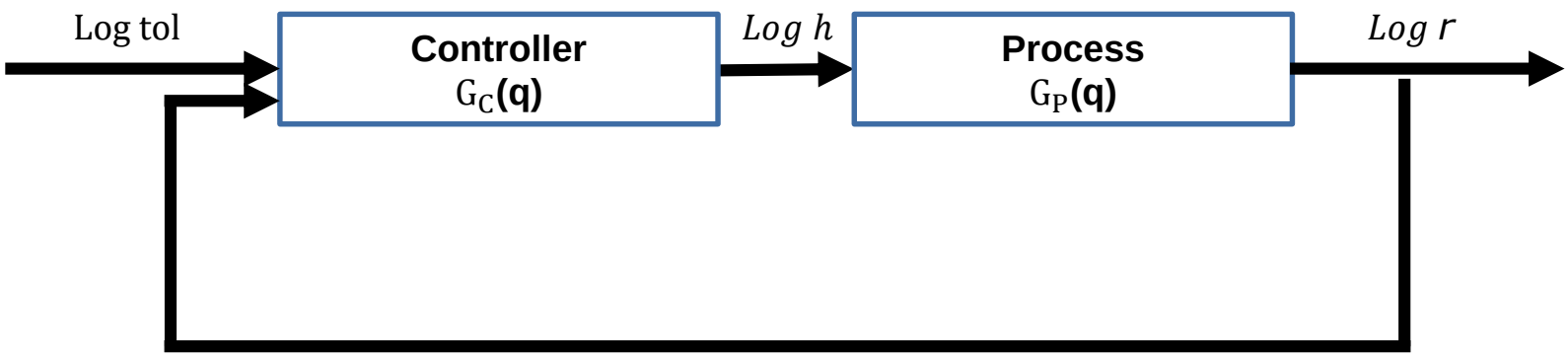
Step Size Controller

Standard Controller

- Tolerance와 error r_n 비율을 이용하여 stepsize h_n 조정
  - $h_n = \gamma(\frac{tol}{r_n})^{\frac{1}{k}} h_{n-1}$

PI Controller

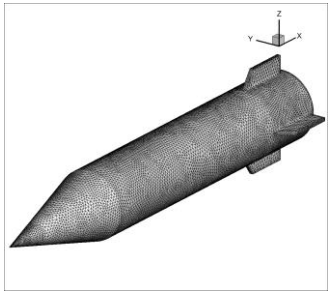
- 기존 방법에 PI 제어 기법 적용
  - $h_n = (\frac{tol}{r_n})^{k_I}(\frac{r_{n-1}}{r_n})^{k_P} h_{n-1}$



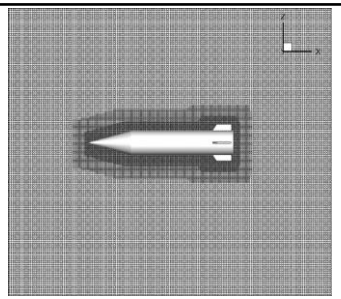
시간 적응 기법

무격자 해석 프로그램

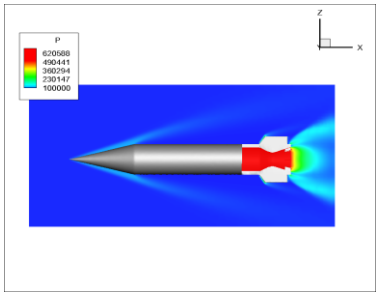
CAD 파일로 부터 표면 격자 자동 생성



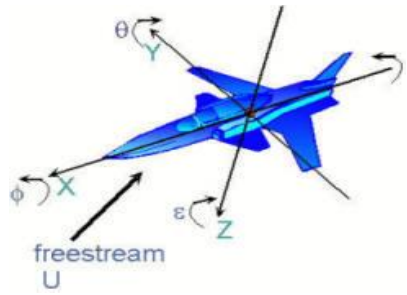
공간 질점 생성 및 Connectivity 구성



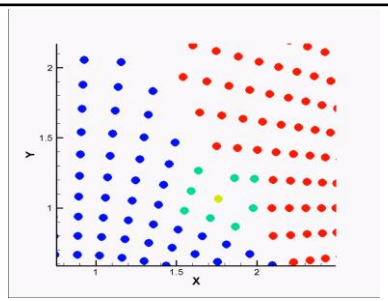
유동 해석 수행



6자유도 Dynamic Eq.으로 움직임 구현

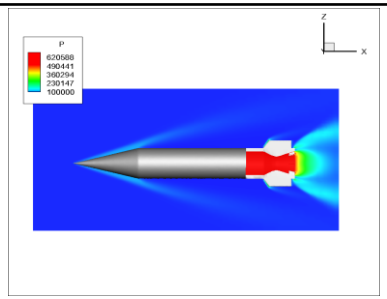


질점 이동 및 Connectivity 재구성

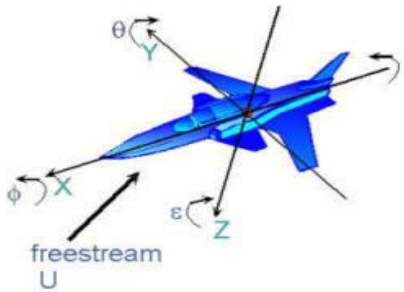


시간 적응 기법

LU-SGS & Dual Time Stepping



6DOF Equation & Runge-Kutta 4th Order



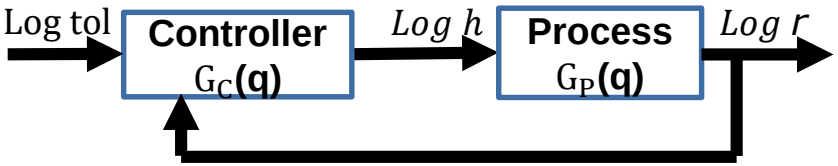
시간 적응 기법 설계

Step Size Control for Dual-Time Stepping

The Closed Loop Using PI Controller

$$h_n = \left(\frac{tol}{r_n}\right)^{k_I}\left(\frac{r_{n-1}}{r_n}\right)^{k_P} h_{n-1}$$

$$k_I = 0.175, \qquad k_P = 0.075$$



Error can be easily defined with any scalar value Ex) ρ, p, T …

$$r_n = \left(\frac{\int(\theta_n-\theta_{n-1})^2 d\Gamma}{\int \theta_n^2 d\Gamma}\right)^{\frac{1}{2}} \qquad \theta = \rho, p, T \ldots$$

Application to Dual-time Stepping

$$\frac{\partial Q}{\partial t} = -R(Q)$$

$$\frac{\partial Q}{\partial \tau} = -\left[\frac{\partial Q}{\partial t} + R(Q)\right] = -R^*(Q)$$

$$\frac{\partial Q^{n+1}}{\partial \tau} = -\frac{1}{h_n}\left[-\frac{pp+2}{pp+1}Q^{n+1} + \frac{pp+1}{pp}Q^n - \frac{1}{pp(pp+1)}Q^{n-1}\right] - R(Q^{n+1}) = R^*(Q), \qquad pp = \frac{h_{n-1}}{h_n}$$

시간 적응 기법 설계

6DOF Equation & Runge-Kutta 4th order method

- **Difficulties in time adaptation application**
  - **Stability region is clearly limited in Runge-Kutta in contrast to LU-SGS**
    - Rigorous selection of PI controller coefficient must be done
  - **No scalar value which can be used in error definition**
    - Analysis for the equation itself must be done
  - **System equation**

$$\begin{bmatrix}\dot{P}\\ \dot{Q}\\ \dot{R}\end{bmatrix} = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{zx}\\ -I_{xy} & I_{yy} & -I_{yz}\\ -I_{zx} & -I_{yz} & I_{zz}\end{bmatrix}^{-1} \begin{bmatrix} \Sigma M_x + I_{yz}(Q^2-R^2) + I_{zx}PQ - I_{xy}RP + (I_{yy}-I_{zz})QR\\ \Sigma M_y + I_{zx}(R^2-P^2) + I_{xy}QR - I_{yz}PQ + (I_{zz}-I_{xx})RP\\ \Sigma M_z + I_{xy}(P^2-Q^2) + I_{yz}RP - I_{zx}QR + (I_{xx}-I_{yy})PQ\end{bmatrix}$$

$$\begin{bmatrix}\dot{U}\\ \dot{V}\\ \dot{W}\end{bmatrix} = \begin{bmatrix} VR - WQ\\ WP - UR\\ UQ - VP\end{bmatrix} + \begin{bmatrix} g_x\\ g_y\\ g_z\end{bmatrix} + \begin{bmatrix} F_x/m\\ F_y/m\\ F_z/m\end{bmatrix}$$

$$\begin{bmatrix}\dot{q}_0\\ q_1\\ q_2\\ q_3\end{bmatrix} = \frac{1}{2}\begin{bmatrix} 0 & -p & -q & -r\\ p & 0 & r & -q\\ q & -r & 0 & p\\ r & q & -p & 0\end{bmatrix}\begin{bmatrix} q_0\\ q_1\\ q_2\\ q_3\end{bmatrix}$$

시간 적응 기법 설계

Step Size Control for Runge-Kutta

M-stage k^{th} Runge-Kutta method

$$Y_1' = f(t_n, y_n), \qquad Y_i' = f\left(t_n + c_i h_n, y_n + h_n \sum a_{ij} Y_i'\right)$$

$$y_{n+1} = y_n + h_n \sum b_j Y_j', \qquad t_{n+1} = t_n + h_n$$

$$e_{n+1} = h_n \sum (b_j - \hat{b}_j) Y_j'$$

$$r_{n+1} = \begin{cases} \|e_{n+1}\| & error\ per\ step(EPS) \\ \dfrac{\|e_{n+1}\|}{h_n} & error\ per\ unit\ step(EPUS) \end{cases}$$

- **Error Analysis**
  - Provide transfer function for Process

If h_n is small,

$$y_{n+1} = P(h_n\lambda) y_n, e_{n+1} = E(h_n\lambda) y_n$$
$$r_{n+1} = \|\phi_n\| h_n^k, \qquad \phi_n = y_n \lambda^{p_e}(\kappa_0 + \kappa_1 h_n \lambda + \cdots)$$

$$log\ r_n = G_{p1}(q) log\ h_n + q^{-1} log\|\phi_n\|$$

$$G_{p1}(q) = kq^{-1}$$

If h_n is increased enough to approach stability region,

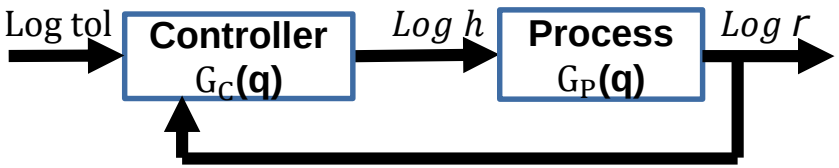
$$e_{n+1} = E(h_n\lambda) y_n = E(h_n\lambda) P(h_{n-1}\lambda) y_{n-1}$$
$$= E(h_n\lambda) \frac{P(h_{n-1}\lambda)}{E(h_{n-1}\lambda)} e_n$$
$$= P(h_s\lambda) \left(\frac{h_n}{h_s}\right)^{C_1} \left(\frac{h_{n-1}}{h_s}\right)^{-C_1+C_2} e_n$$

$$log\ r_n = G_{p2}(q)(log h_n - log h_s), G_{p2}(q) = \frac{C_1 q + C_2 - C_1}{q(q-1)}$$

시간 적응 기법 설계

Step Size Control for Runge-Kutta

The Closed Loop Using Standard Controller



- $h_n = \gamma\left(\frac{tol}{r_n}\right)^{\frac{1}{k}} h_{n-1}$

$logh_n = \frac{1}{k}\frac{q}{q-1}\left(log\left(\gamma^k\, tol\right) - logr_n\right)$ ▶ It can be interpreted as Integral controller with $k_I = \frac{1}{k}$

$$logh_n = G_{c1}(q)(log\ tol - logr_n), \qquad G_{c1}(q) = k_I\frac{q}{q-1}$$

$$logr_n = G_{tol}(q)log\ tol + G_{\phi}(q)log\|\phi_n\|$$

If h_n is small,

$$G_{tol}(q) = \frac{kk_I}{q-1+kk_I}, \qquad G_{\phi}(q) = \frac{q-1}{q(q-1+kk_I)}$$

If h_n is increased,

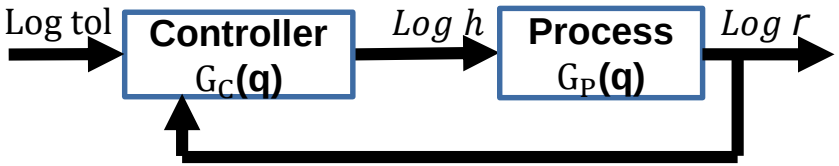
$$G_{tol}(q) = \frac{k_I((C_1-1)q+1-C_1+C_2)}{q^2+(-2+k_I(C_1-1))q+1+k_I(1-C_1+C_2)}$$

$$G_{\phi}(q) = \frac{(q-1)((C_1-1)q+1-C_1+C_2)}{q^2+(-2+k_I(C_1-1))q+1+k_I(1-C_1+C_2)}$$

시간 적응 기법 설계

Step Size Control for Runge-Kutta

The Closed Loop Using PI Controller



- $h_n = (\frac{tol}{r_n})^{k_I}(\frac{r_{n-1}}{r_n})^{k_P} h_{n-1}$

$$logh_n = G_{c2}(q)(log\ tol - logr_n), \qquad G_{c2}(q) = k_I \frac{q}{q-1} + k_P = \frac{(k_I + k_P)q - k_P}{q-1}$$

$$logr_n = G_{tol}(q) log\ tol + G_{\phi}(q) log\|\phi_n\|$$

The denominators are given below

$$G_{p1}(q):\ q^2 + (-1 + k(k_I + k_P))q - kk_P = 0$$

$$G_{p2}(q):\ q^3 + (-2 + C_1(k_I + k_P))q^2 + (1 + C_2(k_I + k_P) - C_1(k_I + 2k_P))q + k_P(C_1 - C_2) = 0$$

$$G_{p3}(q): q^3 + (-2 + (C_1 - 1)(k_I + k_P))q^2 + (1 + C_2(k_I + k_P) + (1 - C_1)(k_I + 2k_P))q + k_P(C_1 - C_2 - 1) = 0$$

$$k_I = \frac{0.3}{k},\ k_P = \frac{0.4}{k}$$

시간 적응 기법 설계

Step Size Control for Runge-Kutta

Error analysis of system equations

$$Y' = \frac{dY}{dt} = F(t;Y), \qquad \textit{with the initial value } Y(a) = Y_0 \ \ (a \le Y \le b)$$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}, F(t;Y) = \begin{bmatrix} f_1(t;y_1,y_2,\dots,y_m) \\ f_2(t;y_1,y_2,\dots,y_m) \\ \vdots \\ f_m(t;y_1,y_2,\dots,y_m) \end{bmatrix}$$

$Y(t)$: *exact solution*
Y_n^*: *theoretical solution*
Y_n: *computed solution*

$$Y_{n+1}^* = Y_n^* + \frac{h}{6}\sum_{j=1}^{4} \gamma_j K_j^*$$

$$K_j^* = F(t_n + \delta_j h, Y_n^* + h\delta_j K_{j-1}^*)$$

$$Y_{n+1} = Y_n + \frac{h}{6}\sum_{j=1}^{4} \gamma_j K_j + R_n$$

$$K_j = F(t_n + \delta_j h, Y_n + h\delta_j K_{j-1})$$

We can get the error by subtracting *exact solution* from *computed solution.*

$$Y_{n+1} - Y(t_{n+1}) = Y_{n+1} - Y(t_n + h)$$

$$= Y_n - Y(t_n) + \frac{h}{6}\sum_{j=1}^{4} \gamma_j (K_j - K(j)) + R$$

$$e_{n+1} = e_n + \frac{h}{6}\sum_{j=1}^{4} \gamma_j (K_j - K(j)) + R$$

시간 적응 기법 설계

Step Size Control for Runge-Kutta

Error analysis of system equations

$$e_{n+1} = e_n + \frac{h}{6}\sum_{j=1}^{4}\gamma_j(K_j - K(j)) + R$$

Assumption 1: $F(t;Y)$ is continuous function of Y in the interval
Assumption 2: Jacobian matrix $(\frac{\partial f_i}{\partial y_j})$ exists in the interval whose endpoints are Y_n and $Y(t_n)$

$$K_1 - K(1) = \left(\frac{\partial f_i}{\partial y_j}\right)e_n$$

$$K_2 - K(2) = \left(\frac{\partial f_i}{\partial y_j}\right)\left\{E + \frac{h}{2}\left(\frac{\partial f_i}{\partial y_j}\right)\right\}e_n$$

$$K_3 - K(3) = \left(\frac{\partial f_i}{\partial y_j}\right)\left\{E + \frac{h}{2}\left(\frac{\partial f_i}{\partial y_j}\right)\left\{E + \frac{h}{2}\left(\frac{\partial f_i}{\partial y_j}\right)\right\}\right\}e_n$$

$$K_4 - K(4) = \left(\frac{\partial f_i}{\partial y_j}\right)\left\{E + h\left(\frac{\partial f_i}{\partial y_j}\right)\left\{E + \frac{h}{2}\left(\frac{\partial f_i}{\partial y_j}\right)\left\{E + \frac{h}{2}\left(\frac{\partial f_i}{\partial y_j}\right)\left\{E + \frac{h}{2}\left(\frac{\partial f_i}{\partial y_j}\right)\right\}\right\}\right\}\right\}e_n$$

시간 적응 기법 설계

Step Size Control for Runge-Kutta

- **Error analysis of system equations**

$$e_{n+1} = \left[E + \frac{h}{6}\left\{\left(\frac{\partial f_i}{\partial y_j}\right) + 2\left(\frac{\partial f_i}{\partial y_j}\right) + 2\left(\frac{\partial f_i}{\partial y_j}\right) + \left(\frac{\partial f_i}{\partial y_j}\right)\right\} + \frac{h^2}{6}\left\{\left(\frac{\partial f_i}{\partial y_j}\right)\left(\frac{\partial f_i}{\partial y_j}\right) + \left(\frac{\partial f_i}{\partial y_j}\right)\left(\frac{\partial f_i}{\partial y_j}\right) + \left(\frac{\partial f_i}{\partial y_j}\right)\left(\frac{\partial f_i}{\partial y_j}\right)\right\}\right.$$

시간 적응 기법 설계

Step Size Control for Runge-Kutta

Error analysis of system equations

$$\hat{e}_{n+1} = \left(\sum_{v=0}^{4} \frac{h^v}{v!} K^v \right) \hat{e}_n \quad with\ K = T^{-1} J T$$

If matrix J is full rank,

Matrix K become diagonal

$$e_{n+1} = \begin{pmatrix} \sum \frac{(h\rho_1)^v}{v!} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sum \frac{(h\rho_m)^v}{v!} \end{pmatrix} \mathrm{e_n}$$

$$\hat{e}_n = diag\left(\left\{ \sum \frac{(h\rho_i)^v}{v!} \right\}^n \right) \times C$$

If matrix J isn't full rank, matrix J cannot be diagonalized.

Some other way must be considered for transformation.

Stability region analysis is based on complex plane.
But, complex properties come from eigenvalues only.

$$matrix\ J \in M_{10}(\mathbb{R})$$

-> matrix K may be transformed to Jordan form

For a matrix on a general scalar field $\mathrm{F}(\mathbb{R})$, Jordan decomposition exists when minimal polynomial is completely decomposed into linear equations in $\mathrm{F}(\mathbb{R})$.

시간 적응 기법 설계

Step Size Control for Runge-Kutta

Error analysis of system equations

- Transformation to Jordan form
  - matrix K has submatrix in the leading diagonal

$$K = \begin{bmatrix} K_{11} & & & & 0 \\ & K_{12} & & & \\ & & \ddots & & \\ & & & K_{ij} & \\ 0 & & & & \ddots \end{bmatrix}, \quad with\ K_{ij} = \begin{bmatrix} \rho_i & 1 & & 0 \\ & \rho_i & \ddots & \\ & & \ddots & 1 \\ 0 & & & \rho_i \end{bmatrix}$$

$$The\ number\ of\ K_{ij} = The\ number\ of\ independent\ eigenvalues$$

$$K_{ij} \in M_l(\mathbb{C}), l = algebric\ multiplicity$$

$$\hat{e}_{n+1} = \left(\sum_{v=0}^{4} \frac{h^v}{v!} K_{\sigma\gamma}^v \right) \hat{e}_n, \qquad with\ K_{\sigma\gamma}^v = \begin{bmatrix} \rho_\sigma^v & \binom{v}{1} \rho_\sigma^{v-1} & \cdots & \binom{v}{\gamma-1} \rho_\sigma^{v-\gamma+1} \\ 0 & \rho_\sigma^v & \cdots & \binom{v}{\gamma-2} \rho_\sigma^{v-\gamma+2} \\ & & \ddots & \\ 0 & & & \rho_\sigma^v \end{bmatrix}$$

시간 적응 기법 설계

◆ Step Size Control for Runge-Kutta

■ Error analysis of system equations

- Transformation to Jordan form(contd.)

$$\hat{e}_{n+1} = \begin{bmatrix} \sum_{v=0}^{4}\frac{(h\rho_\sigma)^v}{v!} & \frac{h}{1!}\sum_{v=0}^{3}\frac{(h\rho_\sigma)^v}{v!} & \frac{h^2}{2!}\sum_{v=0}^{2}\frac{(h\rho_\sigma)^v}{v!} & & \frac{h^4}{4!} & \cdots & 0 \\ 0 & \sum_{v=0}^{4}\frac{(h\rho_\sigma)^v}{v!} & & \cdots & & & \\ 0 & 0 & \sum_{v=0}^{4}\frac{(h\rho_\sigma)^v}{v!} & & & & \\ & \vdots & & \ddots & & \vdots & \\ & & & \cdots & & \sum_{v=0}^{4}\frac{(h\rho_\sigma)^v}{v!} & \\ 0 & & & & & & \sum_{v=0}^{4}\frac{(h\rho_\sigma)^v}{v!} \end{bmatrix} \hat{e}_n$$

$$\boxed{\hat{e}_{i,(n+1)} = \sum_{j=0}^{\gamma-1}\frac{h^j}{j!}\sum_{v=0}^{4-j}\frac{(h\rho_\sigma)^v}{v!}\hat{e}_{(i+j),n} \quad with\ (i = 1, 2, \ldots, \gamma; 0 \le j \le 4)}$$

시간 적응 기법 설계

Step Size Control for Runge-Kutta

Error analysis of system equations

$$\hat{\boldsymbol{e}}_{\boldsymbol{n+1}} = \left(\sum_{\boldsymbol{v=0}}^{\boldsymbol{4}} \frac{\boldsymbol{h}^{\boldsymbol{v}}}{\boldsymbol{v}!} \boldsymbol{K}^{\boldsymbol{v}} \right) \hat{\boldsymbol{e}}_{\boldsymbol{n}} + \hat{\boldsymbol{R}}$$

- **Homogeneous solution**
  - $\hat{e}_n = diag\left(\left\{\sum \frac{(h\rho_1)^v}{v!}\right\}^n\right) \times C$
  - $\hat{e}_{i,(n+1)} = \sum_{j=0}^{\gamma-1} \frac{h^j}{j!} \sum_{v=0}^{4-j} \frac{(h\rho_\sigma)^v}{v!} \widehat{e}_{(i+j),n} \quad with\ (i = 1,2,\ldots,\gamma; 0 \le j \le 4)$
- **Particular solution**

$$\left(\boldsymbol{E} - \sum_{\boldsymbol{v=0}}^{\boldsymbol{4}} \frac{\boldsymbol{h}^{\boldsymbol{v}}}{\boldsymbol{v}!} \boldsymbol{J}^{\boldsymbol{v}} \right) \boldsymbol{e}_{\boldsymbol{n}} = \boldsymbol{R}$$

$$\boldsymbol{e}_{\boldsymbol{n}} = \left(\boldsymbol{E} - \sum_{\boldsymbol{v=0}}^{\boldsymbol{4}} \frac{\boldsymbol{h}^{\boldsymbol{v}}}{\boldsymbol{v}!} \boldsymbol{J}^{\boldsymbol{v}} \right)^{-1} \boldsymbol{R}$$

$$\therefore \boldsymbol{e}_{\boldsymbol{n}} = \mathrm{T} \begin{bmatrix} \mathrm{c}_{11} & & 0 \\ & \ddots & \\ 0 & & \mathrm{c}_{\mathrm{mm}} \end{bmatrix} \begin{bmatrix} \left\{\sum \frac{(h\rho_1)^v}{v!}\right\}^n \\ \vdots \\ \left\{\sum \frac{(h\rho_{\mathrm{m}})^v}{v!}\right\}^n \end{bmatrix} - \left(\sum_{\boldsymbol{v=1}}^{\boldsymbol{4}} \frac{\boldsymbol{h}^{\boldsymbol{v}}}{\boldsymbol{v}!} \boldsymbol{J}^{\boldsymbol{v}} \right)^{-1} \boldsymbol{R}$$

시간 적응 기법 설계

◆ Step Size Control for Runge-Kutta

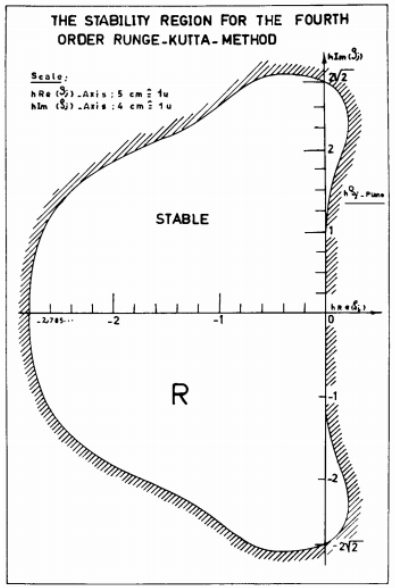
■ Error analysis of system equations

$$\boldsymbol{e_n} = \mathrm{T}\begin{bmatrix} \mathrm{c}_{11} & & 0 \\ & \ddots & \\ 0 & & \mathrm{c}_{\mathrm{mm}} \end{bmatrix}\begin{bmatrix} \left\{\sum \frac{(h\rho_1)^v}{v!}\right\}^n \\ \vdots \\ \left\{\sum \frac{(h\rho_{\mathrm{m}})^v}{v!}\right\}^n \end{bmatrix} - \left(\sum_{v=1}^{4} \frac{\boldsymbol{h}^v}{\boldsymbol{v}!}\boldsymbol{J}^v\right)^{-1} \boldsymbol{R}$$

- **Runge-Kutta 4th order method for system equations is said to be stable when $\left\{\sum \frac{(h\rho_i)^v}{v!}\right\}$ is in the unit circle in the complex plane.**

$$\left|\sum \frac{(h\rho_i)^v}{v!}\right| < 1$$

- **Stability region**
  - **Stability region of RK4 for system equations is similar to that for the 1D ODE.**



시간 적응 기법 설계

◆ Step Size Control for Runge-Kutta

■ Error analysis of system equations

$$\boldsymbol{e_n} = \mathrm{T}\begin{bmatrix} \mathrm{c}_{11} & & 0 \\ & \ddots & \\ 0 & & \mathrm{c}_{\mathrm{mm}} \end{bmatrix}\begin{bmatrix} \left\{\sum \frac{(h\rho_1)^v}{v!}\right\}^n \\ \vdots \\ \left\{\sum \frac{(h\rho_{\mathrm{m}})^v}{v!}\right\}^n \end{bmatrix} - \left(\sum_{v=1}^{4} \frac{\boldsymbol{h}^v}{\boldsymbol{v}!}\boldsymbol{J}^v\right)^{-1} \boldsymbol{R}$$

$$i\ corresponding\ to\ the\ maximum\ error(e_{max}) \neq i\ that\ maximizes \left|\sum \frac{(h\rho)^v}{v!}\right|$$

1. It is mathematically reasonable to control the stepsize with r_n which maximizes $\left|\sum \frac{(h\rho)^v}{v!}\right|$, rather than based on e_{max}

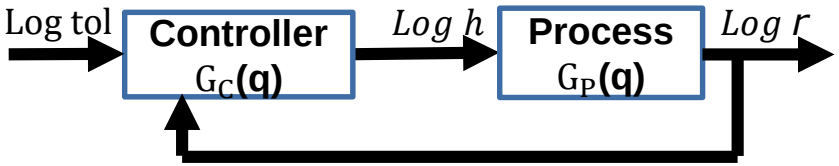
2. Process transfer function for 1D ODE is identical for system equations

=> Controller coefficient $\mathrm{k_I}, \mathrm{k_P}$ for 1D ODE can be used.

시간 적응 기법 설계

Determining Controller for Stepsize Control

The Closed Loop Using PI Controller



- $h_n = (\frac{tol}{r_n})^{k_I}(\frac{r_{n-1}}{r_n})^{k_P} h_{n-1}$

LU-SGS : $k_I = 0.175,\ k_P = 0.075$

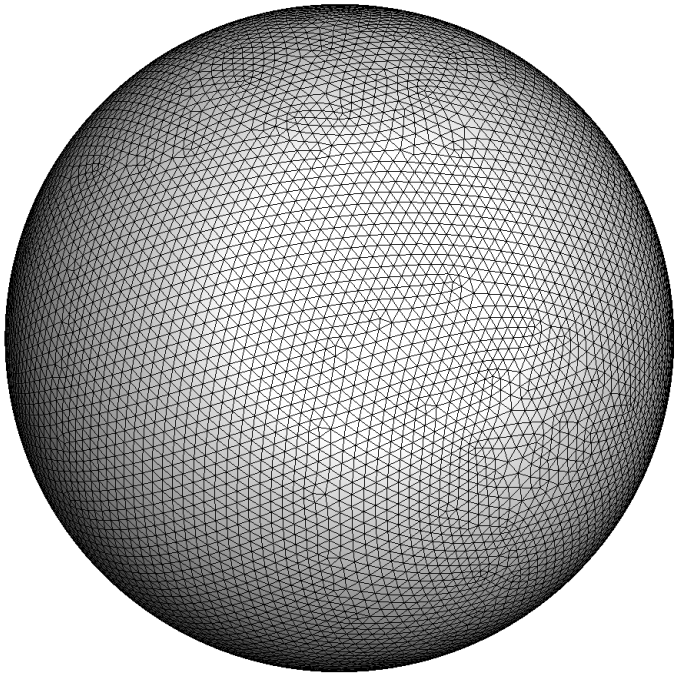
Runge-Kutta method : $k_I = \frac{0.3}{k}$, $k_P = \frac{0.4}{k}$ $(k = O(leading\ error))$

- Prevent unnecessary oscillation

$$\frac{h_n}{h_{n-1}} = \begin{cases} 2 & (\frac{h_n}{h_{n-1}} \geq 2) \\ \frac{h_n}{h_{n-1}} & ELSE \\ 1 & (1 \leq \frac{h_n}{h_{n-1}} \leq 1.2) \end{cases}$$

시간 적응 기법

- Sphere 해석에서의 시간 적응 기법
  - 해석 방정식 : Euler Equation
  - 공간 차분 : M-AUSMPW+
  - 공간 내삽 : 3rd MLP
  - 시간 적분 : LU-SGS & Dual Time Stepping

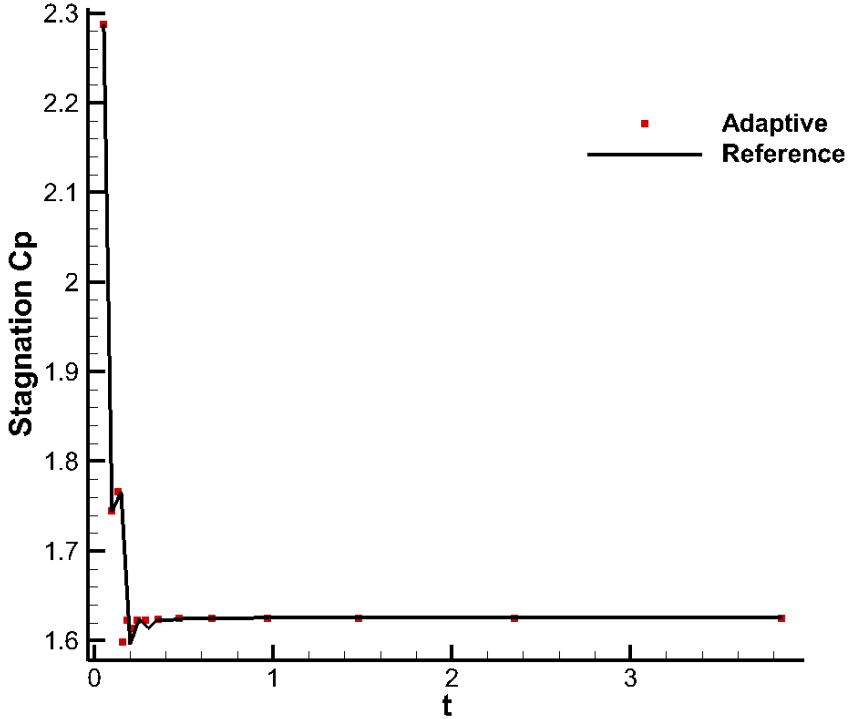


Operation condition	AOA (Deg)	자유류 속도 (Ma.)	온도(K)
	0.0	2.0	300.0

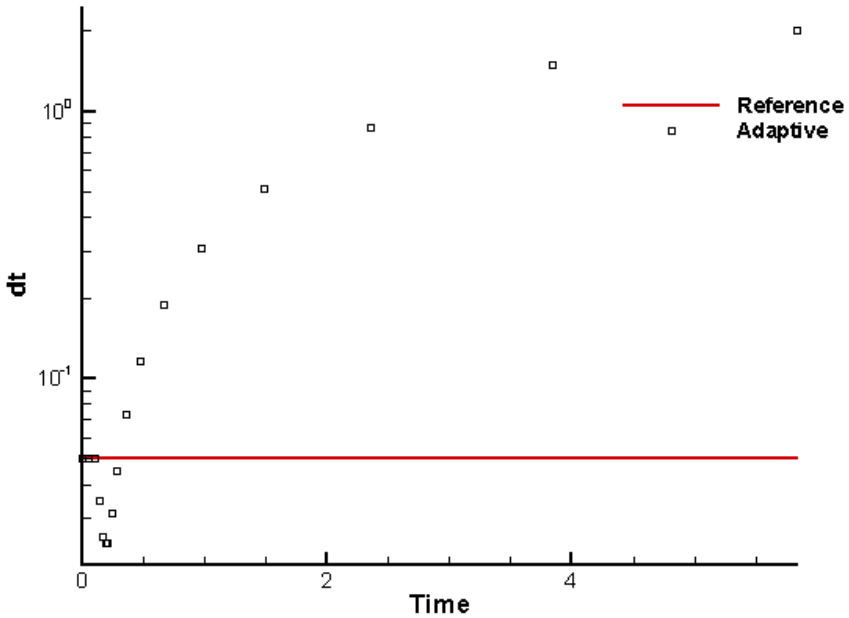
$$tolerance = 10^{-3} \ \& \ Starting\ DT = 0.05$$

시간 적응 기법

◆ Sphere 해석에서의 시간 적응 기법



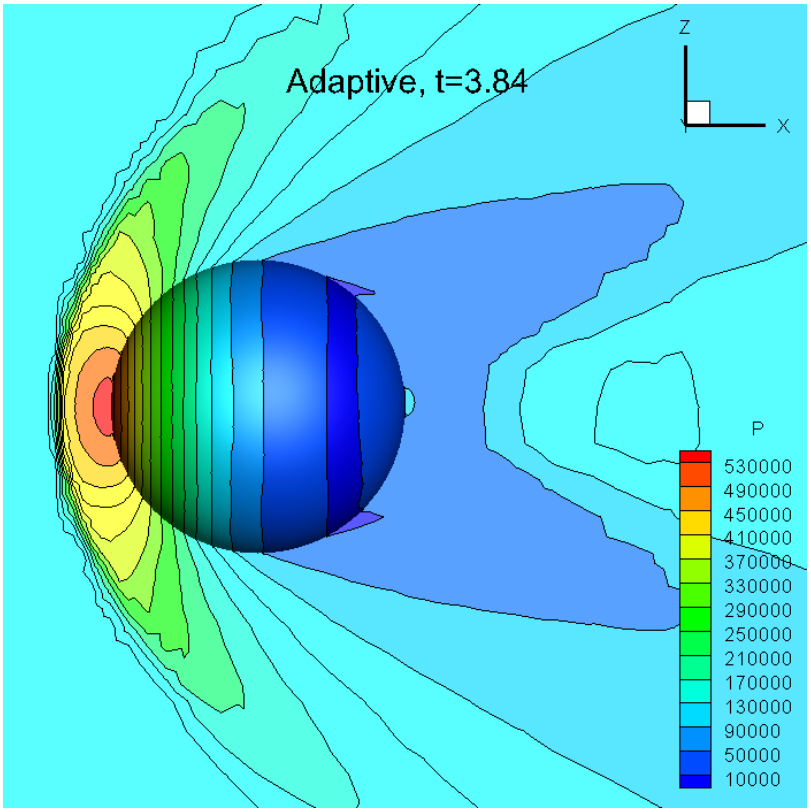
Stagnation Point Pressure Coefficient



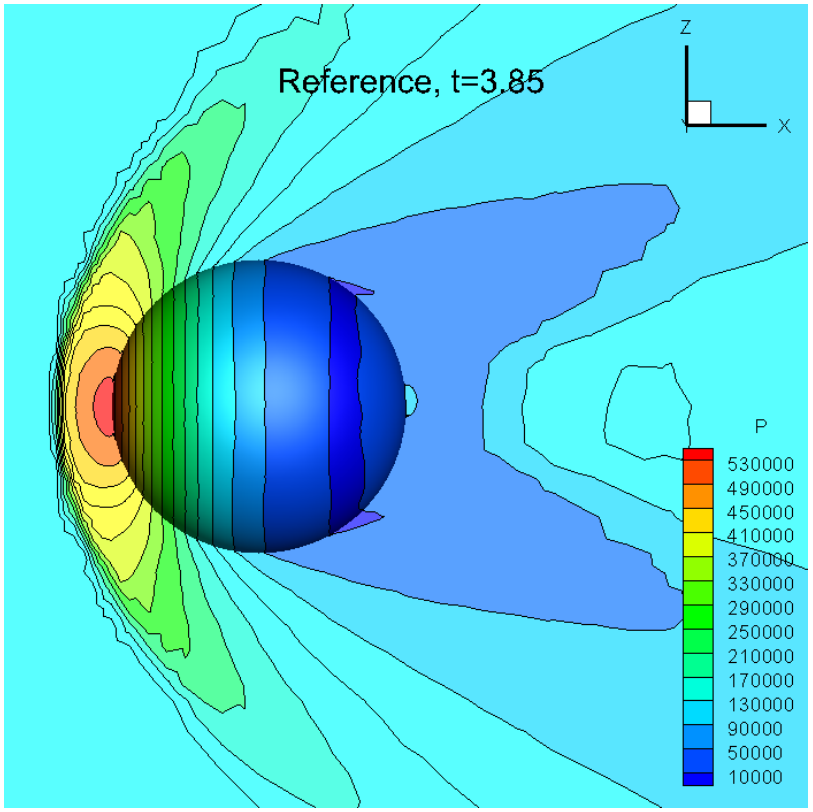
Physical Time Step Change

시간 적응 기법

◆ **Sphere 해석에서의 시간 적응 기법**



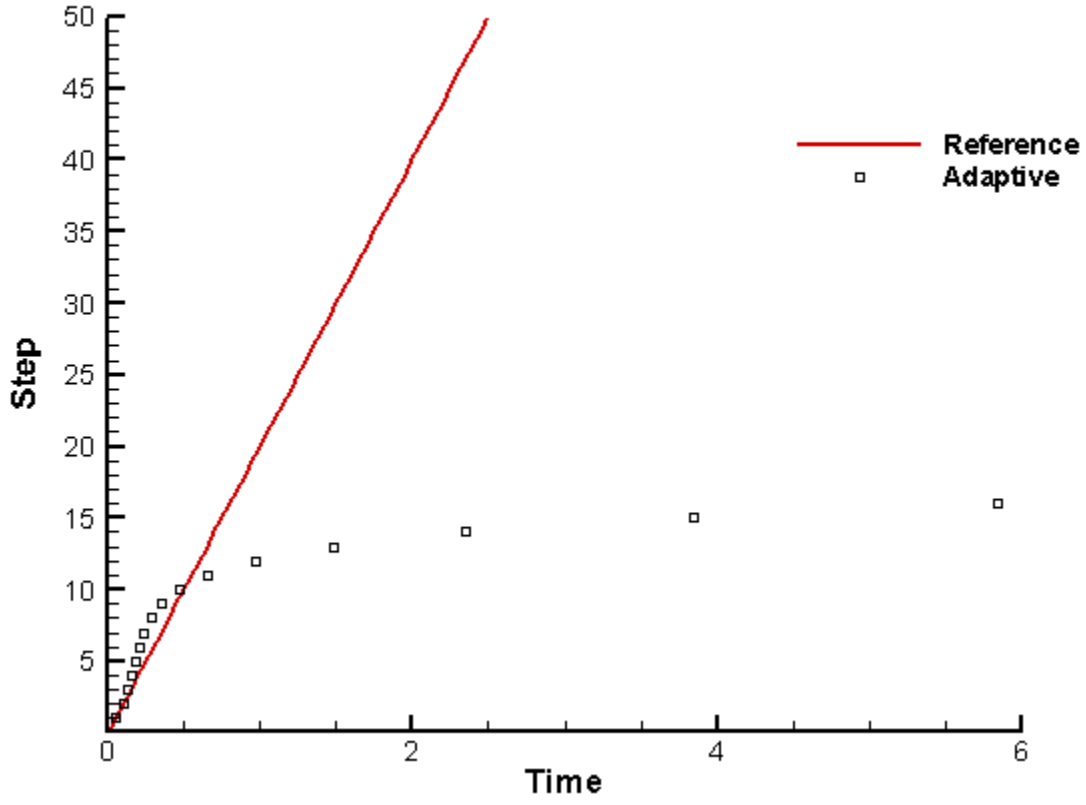
Pressure Contour
[Adaptive]



Pressure Contour
[Non-Adaptive]

시간 적응 기법

◆ Sphere 해석에서의 시간 적응 기법



The Number of Calculated Timestep

16 Step .vs. 114 Step
- 약 7배 정도 효율적

목 차

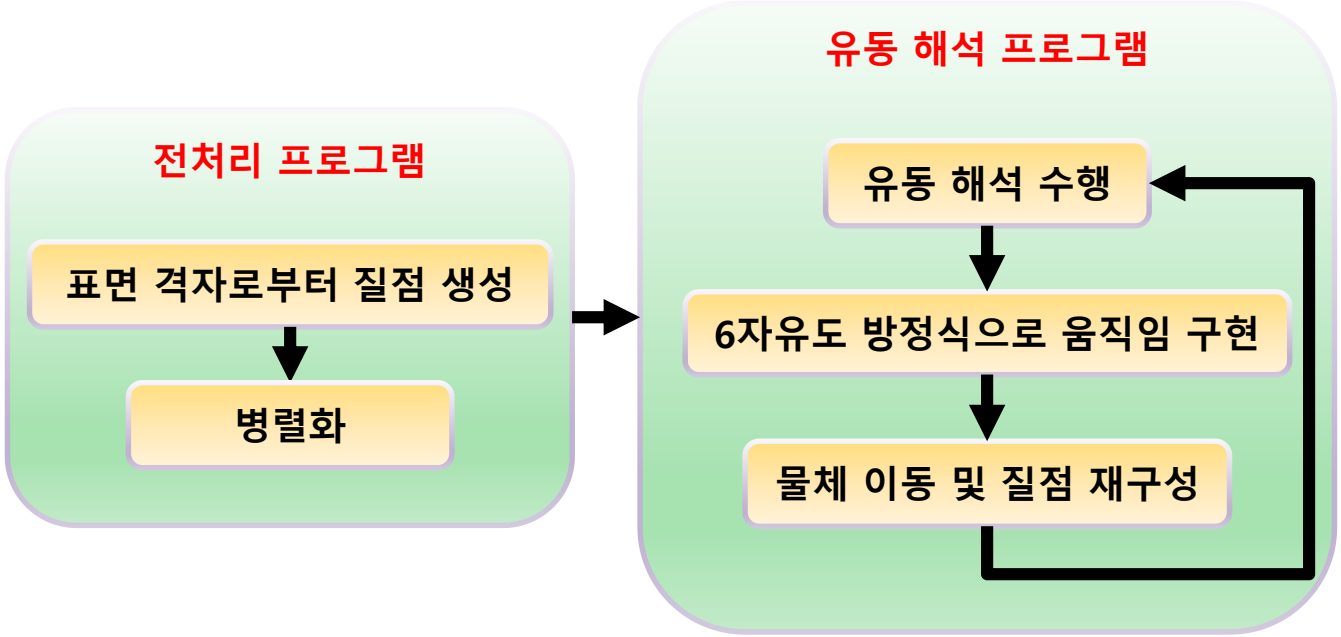
결론

- **Summary**
  - **Spatial adaptation algorithm is developed.**
  - **Time adaptation algorithm**
    - **Application to LU-SGS & Dual time stepping**
    - **Runge-Kutta 4th order method**
      - **Error analysis & PI controller design**

감사합니다

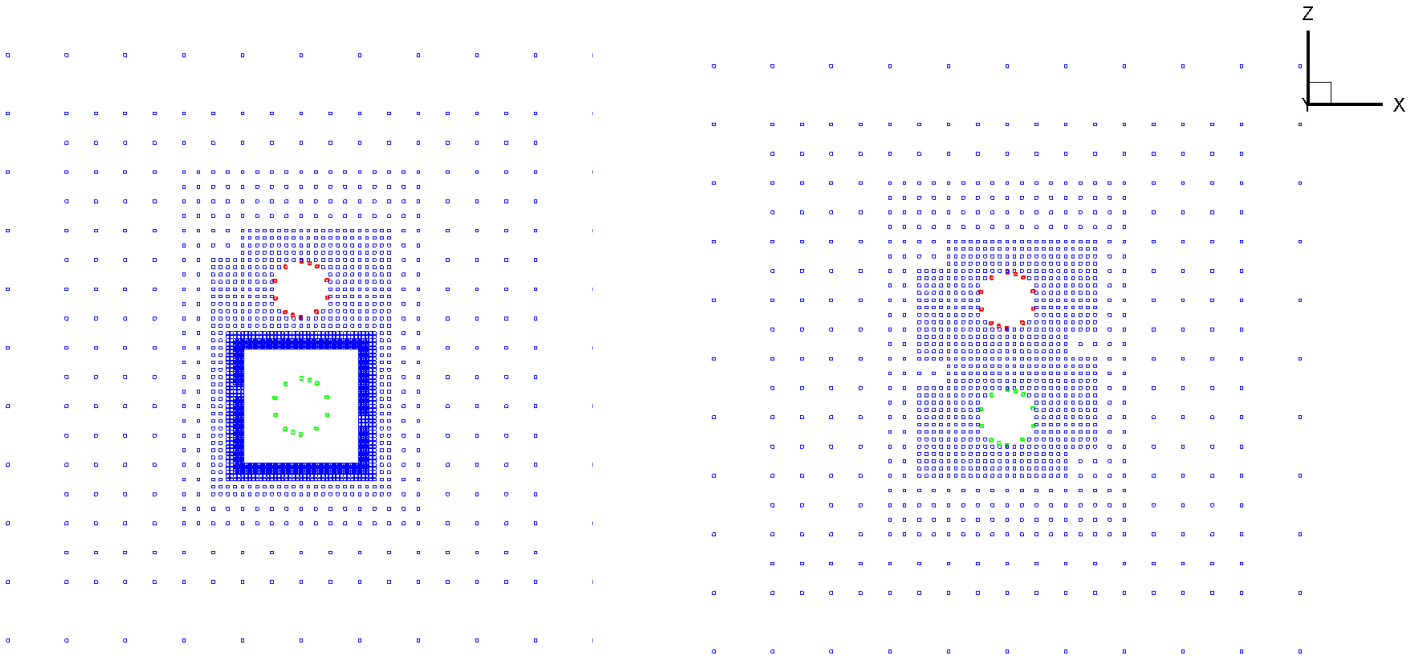
공간 질점계 적응 기법

◆ 배경 질점계 구조 변경



Fixed – Multi objects

- **In/Out Determination**
  - **Bug fixed**

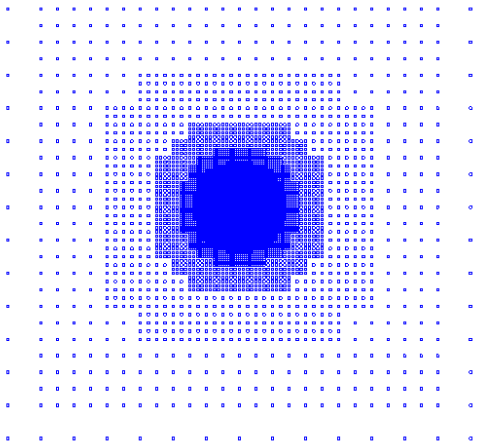


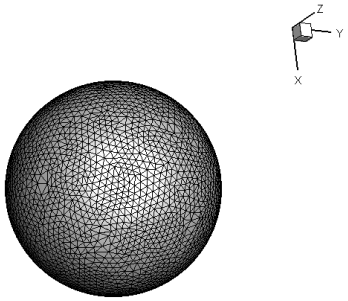
개선2 – New In/Out Determination

◆ Octree 분할

- 유동 해석을 위한 질점은 표면 근처에서 많이 분포되어 있어야 함.
  - ▶ Octree cell 크기가 표면 격자와 같은 크기가 되는 level까지 분할
- 물체 표면에서 Octree를 더 많이 분할하기 위해 Cell Type 결정 필요
  - Inside Cell Type – 물체 내부에 있는 Octree Cell
  - Outside Cell Type – 물체 외부에 있는 Octree Cell
  - Cut Cell Type – 물체 표면에 걸쳐있는 Octree Cell ▶ Cut Cell을 연속적으로 분할

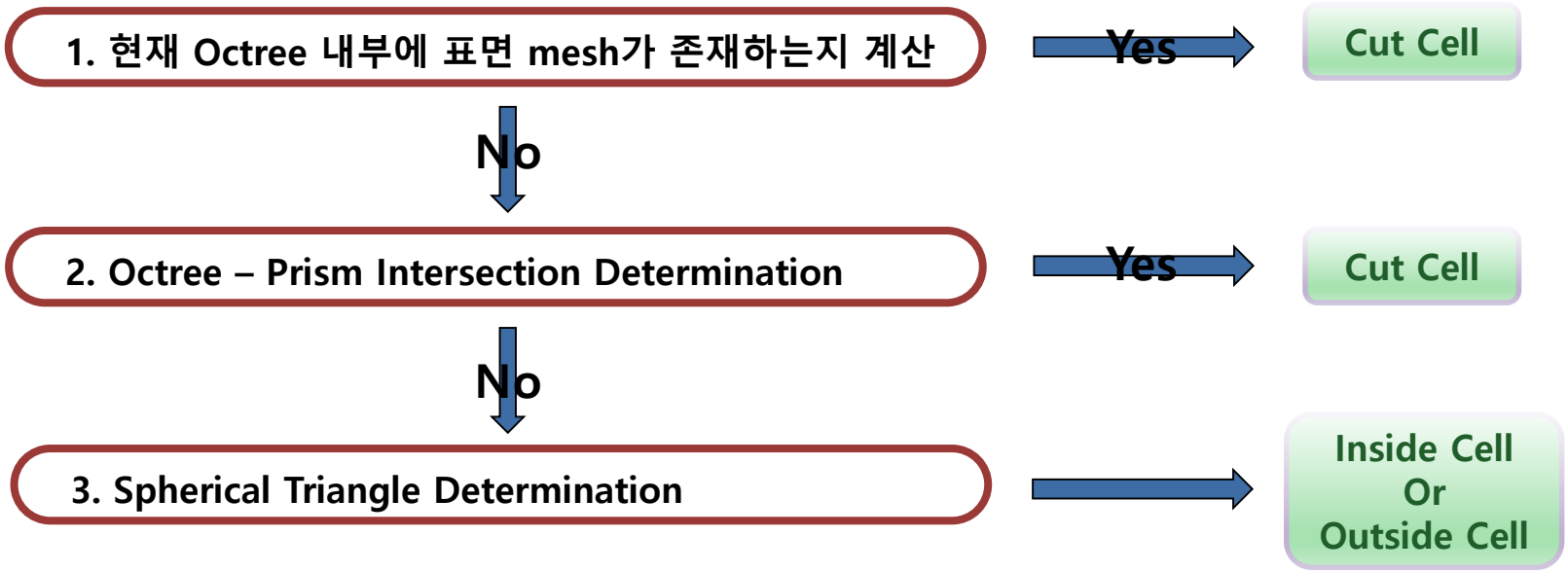
- Cell Type을 결정하기 위해 물체 내,외부 판단을 해야함
  - ▶ 기존 방법은 판단할 때마다 모든 표면 격자에 대해 계산 – 많은 시간 소요





개선2 – New In/Out Determination

- **Octree 분할**
  - 내,외부 판단 알고리즘 개선
    - 기존 알고리즘
      - 판단 1번을 위해 모든 표면 격자에 대한 계산 3번 필요
        - 표면 격자가 많은 경우 & Octree 점이 많은 경우 ▶ 계산 시간이 기하급수적으로 증가



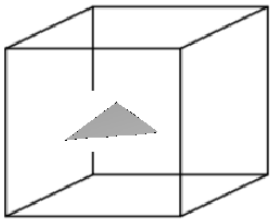
개선2 – New In/Out Determination

◆ Octree 분할

■ 내,외부 판단 알고리즘 개선

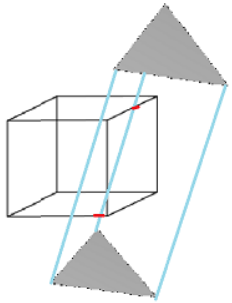
- 기존 알고리즘

1. 현재 Octree 내부에 표면 mesh가 존재하는지 계산



- 현재 판단 대상이 되는 Cell의 좌표와 모든 표면 mesh의 좌표를 비교
  - 현재 Cell 내부에 표면 mesh가 존재하면 Cut Cell

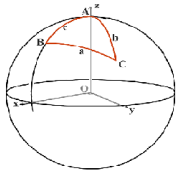
2. Octree – Prism Intersection Determination



- Cell 내부에 실제로 표면 mesh가 들어있지 않더라도 Cut Cell 파악
  - Prism끼리 연결한 직선이 Cell과 겹치는지 계산
  - Intersection되면 Cut Cell

3. Spherical Triangle Determination

- Cut Cell이 아닐 경우 내, 외부 판단
  - 각각의 표면 mesh마다 반지름이 1인 구에 대한 spherical area 계산
  - 모든 표면 mesh에 대한 spherical area 합으로 내,외부 판단



  - 이론적으로, 합이 4*pi면 내부, 0이면 외부

개선2 – New In/Out Determination

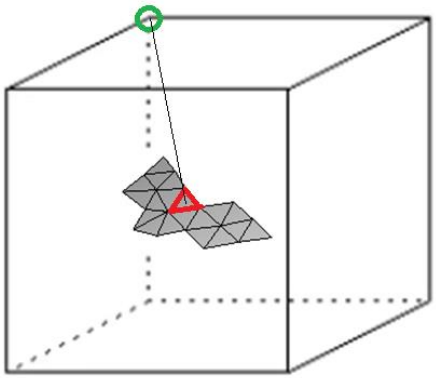
◆ Octree 분할

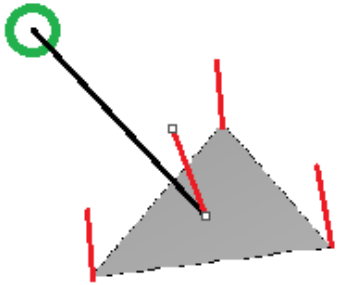
■ 내,외부 판단 알고리즘 개선

- 새로운 알고리즘1

1. Octree 내부에 있는 표면 격자 찾기
2. 검사하고자 하는 점에 대해 가장 가까운 거리의 표면 격자 계산
3. 내적으로 내,외부 판단
4. Cell type 결정

8개 꼭지점에 대해 반복





1. Octree 내부에 있는 표면 격자에 대해서만 계산 (모든 표면 격자 X)
2. 1번의 판단으로 Cell type 결정 (3번 X)

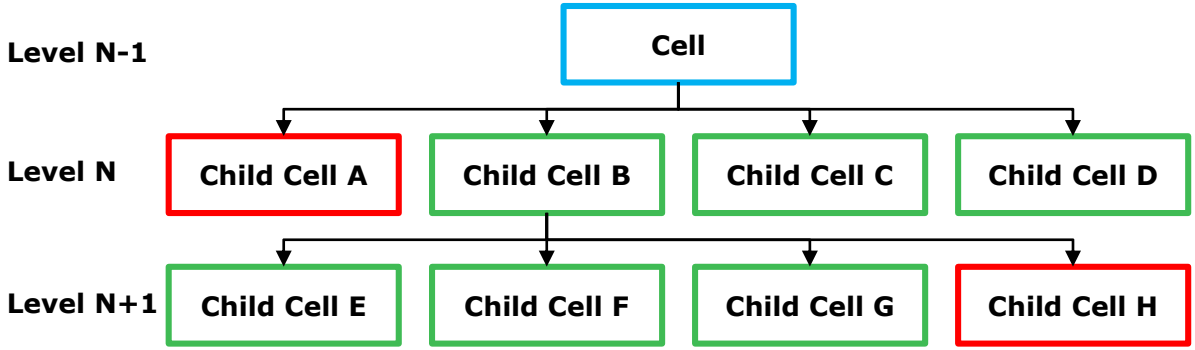
▶ 시간 단축. But, Failure!!

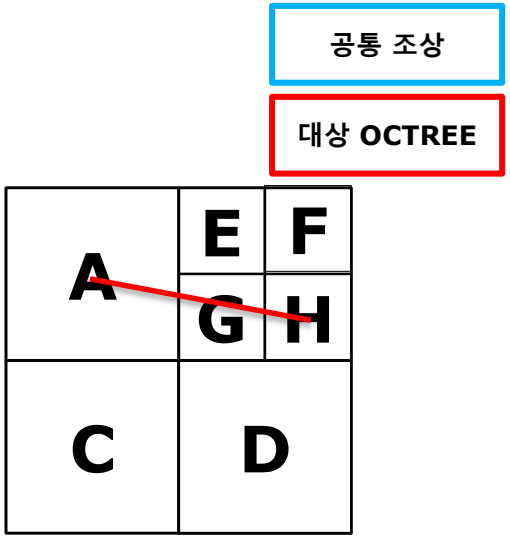
개선2 – New In/Out Determination

◆ Octree 분할

- 내,외부 판단 알고리즘 개선
  - 새로운 알고리즘2

1. 가장 가까운 in/out cell 찾기
2. 공통 조상 찾기
3. 공통 조상 내부에 있는 표면에 대해 ray casting
4. Cell type 결정

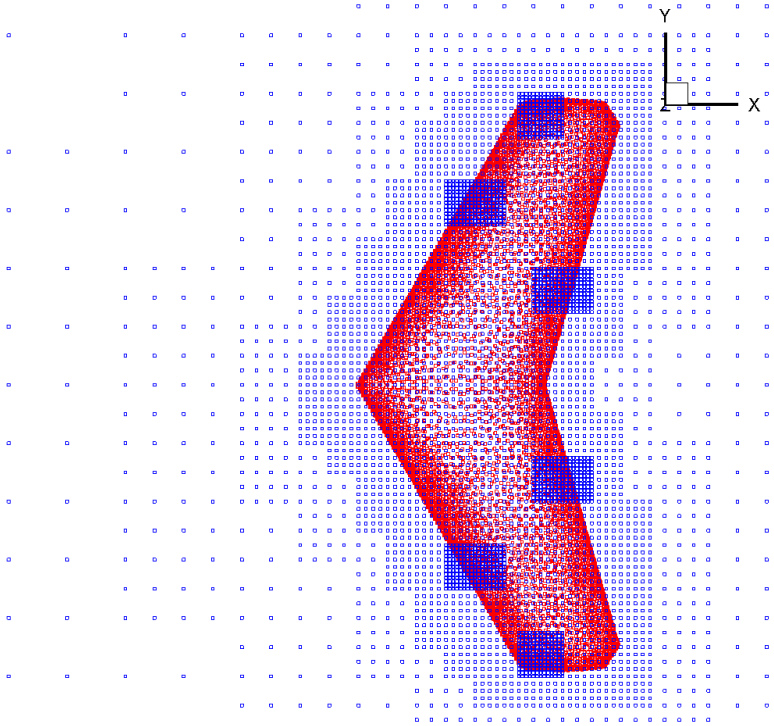


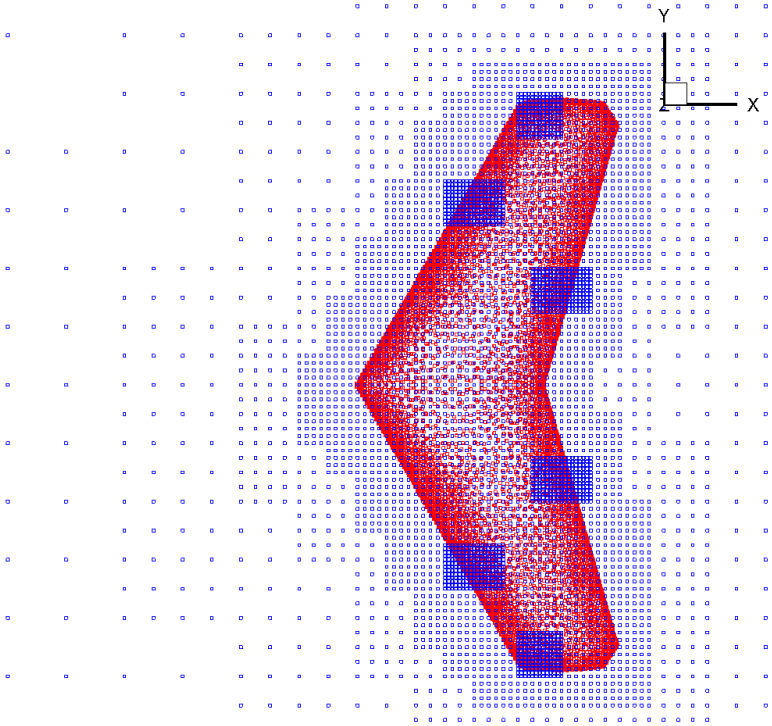


▶ 시간 단축. And, Robustness!!

Octree-Symmetric division

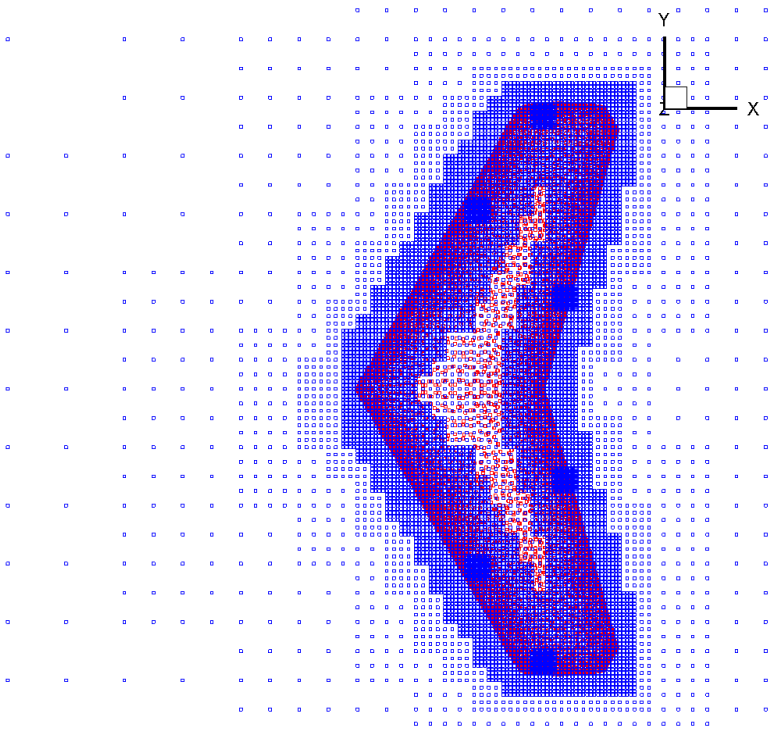
- **In/Out Determination**
  - **기존 내외부, New 내외부**

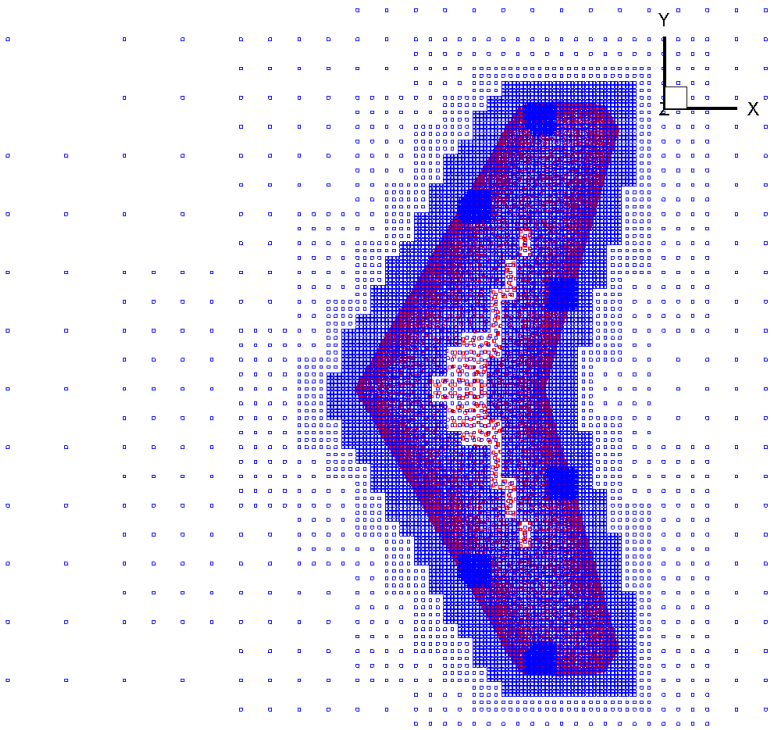




Octree-Symmetric division

- **In/Out Determination**
  - **기존 내외부, New 내외부**

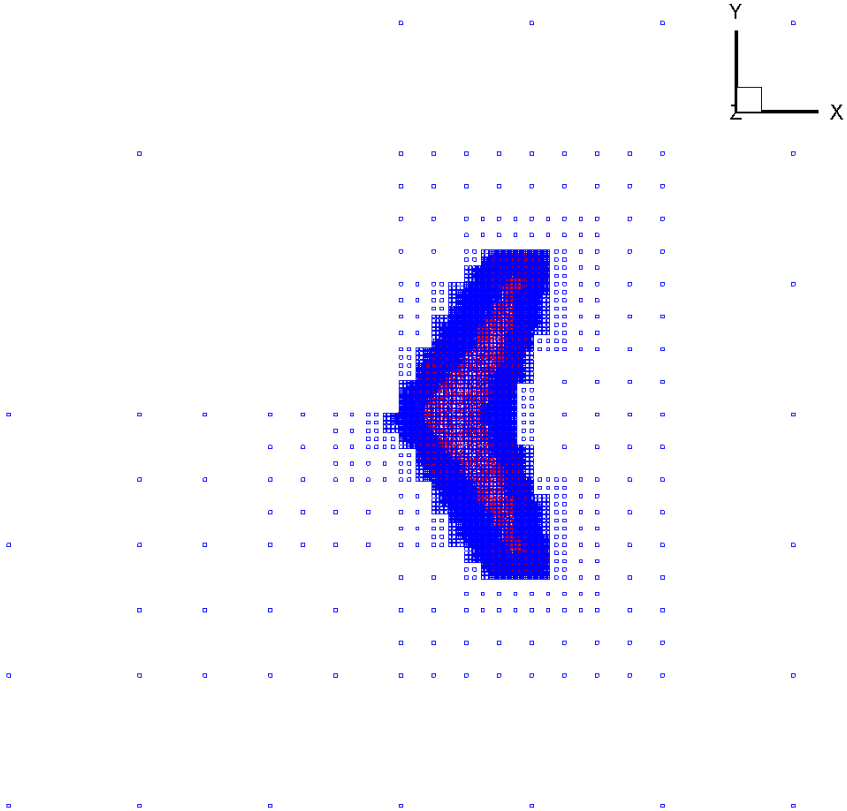


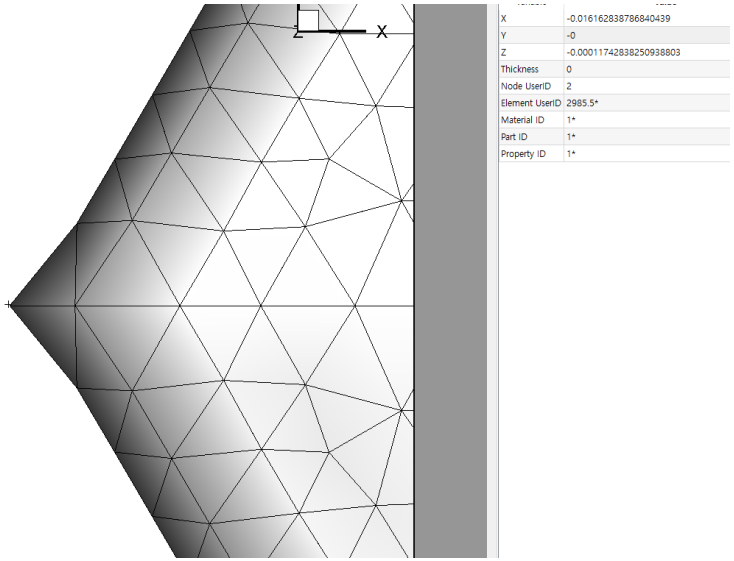


Octree-Symmetric division

In/Out Determination

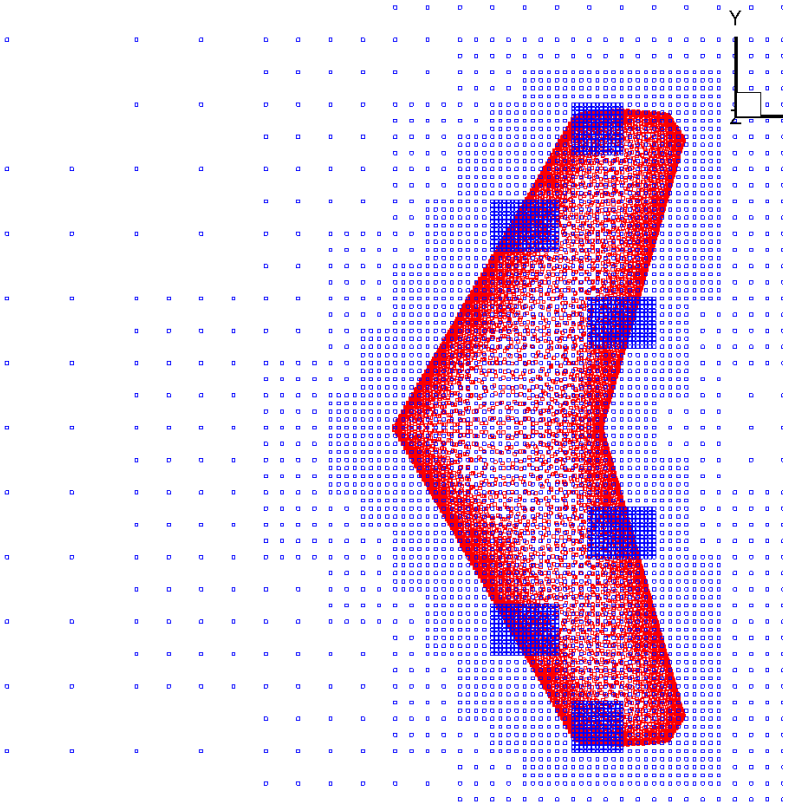
- 극단적인 Input을 주었을 때는 다음과 같은 결과

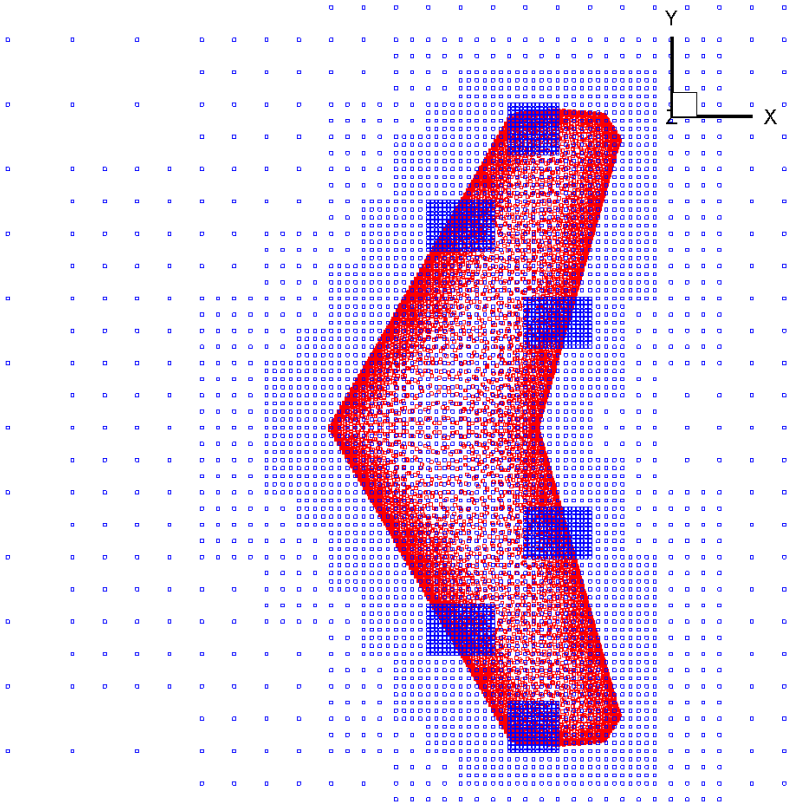




Octree-Symmetric division

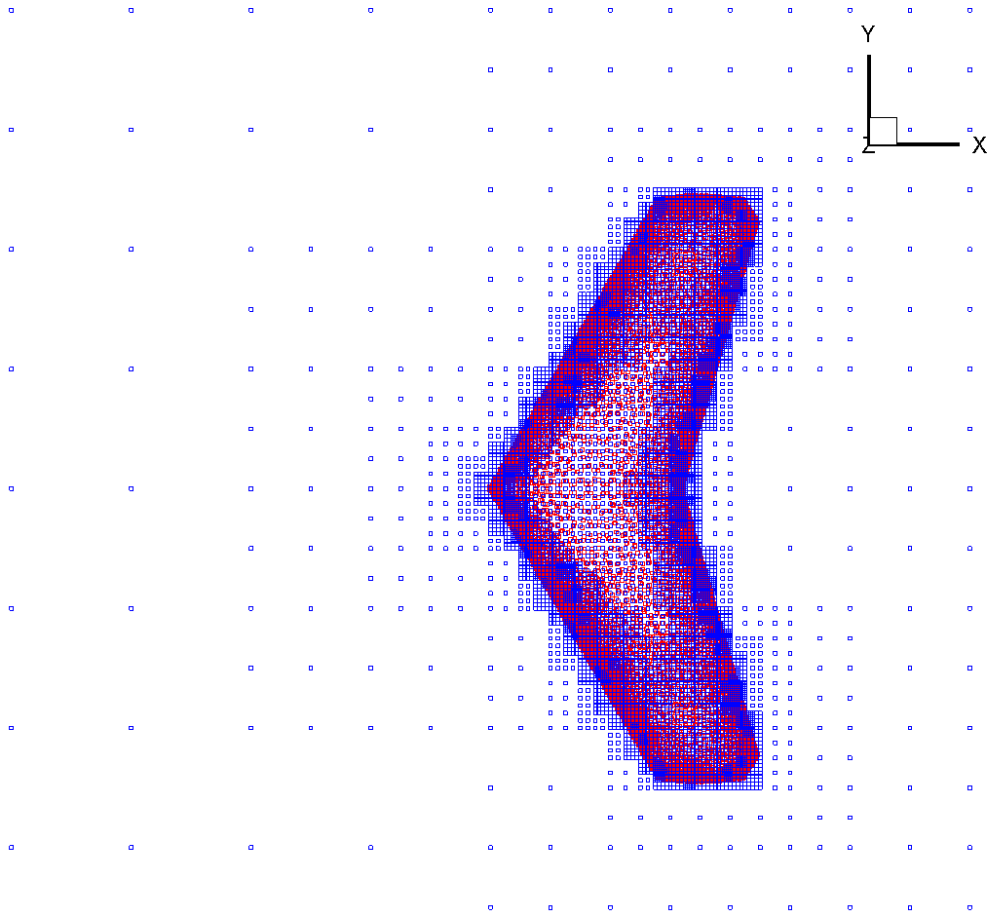
- **In/Out Determination**
  - **Bug fixed**





Octree-Symmetric division

- **In/Out Determination**
  - **Bug fixed**



Octree-Symmetric division

◆ Shift & Smooth Division